

On Negotiating Automatic Device Configuration in Smart Environments

Kay Connelly and Ashraf Khalil
Indiana University
{connelly, akhalil}@cs.indiana.edu

Abstract

As the number of mobile devices we carry grows, the job of managing those devices throughout the day becomes cumbersome. In addition, the ability to use mobile devices in unethical ways without being detected is becoming widespread [1, 2]. It is desirable for mobile devices to automatically configure themselves based on the context of the environment and user preferences for both convenience and security purposes. Having the cell phone switch the ringer off and turn on the vibrate notification in a movie theatre is convenient to the user; while having the video phone disable its video capabilities in a locker room is necessary for others' privacy. We present a preliminary architecture for automatic device configuration in smart environments and explore the issues in negotiating between the device owner's wishes and those of the owner of the smart environment.

1. Introduction

Mobile and wearable computational devices are becoming widespread. Beyond laptops and PDAs, intelligent devices such as watches, active badges, cell phones and mp3 players are beginning to interact with our environments as part of our everyday lives. For example, IBM's Linux watch can be used as an authentication device [3], whereas the active badge can be used to locate resources close to the badge wearer [4].

How we use such devices often changes depending on our social context. For example, a user may want their mp3 player to turn off once they enter an office, or their cell phone ringer to be disabled during an important meeting. As the number of devices we carry with us grows, managing such devices throughout the day as our context changes can become overwhelming.

In order to make device management less cumbersome, and thus the devices more usable, devices should be able to automatically reconfigure themselves based on the current context and user preferences. In addition, as abuse of ubiquitous computing devices becomes more prevalent

[1, 2], systems that can in some sense force configuration changes on devices become interesting. For example, following legislation that outlaws the use of video phones in locker rooms, a gym may want the ability to temporarily turn off the camera capabilities of a video phone. Similarly, a university may wish to disable all cell phone calls (except for an outgoing 911) in a lecture hall when an exam is scheduled.

In this paper, we describe a preliminary architecture for automatic device configuration. In section 2, we review the related work. In section 3, we give a detailed motivating example based on cell phone usage. The design of our system is presented in Section 4. We discuss the various mechanisms to resolve conflicts between device and space owners in Section 5. We present our current status and future plans in Section 6 and conclude in Section 7.

2. Related work

Some existing projects on automatic configuration focus on installing, building and augmenting running application with new services at run time [5, 6]. Our work focuses on altering a device's configuration according to the context of the physical space and the device owner's preferences. We address issues such as automatic policy conflict resolution, trust between the space and the devices, and control versus convenience.

The work of Gaia and Interactive Workspaces projects [7, 8] support automatic integration of devices in the smart, interactive space. Mobile devices automatically register themselves within the space. Most smart environments focus on integrating the services offered by mobile devices with the space services and vice versa. Although smart environments offer automatic integration of the devices, they do not try to enforce certain usage policies or to agree on what is allowed and not allowed within that space.

Other work in the field of the interaction between devices and the environment around them focuses on empowering a certain type of device to be more environment-aware in order to adapt their behavior

accordingly. An example of this is the SenSay project [9] where the cell phone is augmented with sensors and other sources of information to sense the context of the user and adapt the cell phone functionality accordingly. This work focuses on user convenience without taking into account the environment policy. Similar works that focus on specific classes of devices without taking into consideration the policy of the surrounding environment can be found in [10].

3. Motivating example

While our system works with any mobile device, we will use cell phones as our motivating example. Based on social context, the user may want their cell phone to behave in different ways. For example, when a movie starts in a theatre, a user may want their ringer to be disabled and the vibrate notification turned on. Once they leave the theatre, they would like the phone to return to its normal configuration with the ringer enabled. Alternatively, the user may not wish to be disturbed at all and instead have the cell phone take messages and notify the user of the messages once the movie is over. With the combination of caller ID, a user’s configuration preferences can become arbitrarily complex based on the importance of the caller to the user.

Aside from user preferences, the smart space environment may have its own policies when it comes to devices brought into that space. The theatre, for example, could have a policy that cell phone ringers are not allowed to ring, as that would disturb their other customers. Determining when a space owner’s wishes override an individual user’s preferences is an interesting societal question. However, there are certainly some cases which will be driven by law. As abuses such as the unauthorized taking of pictures of undressed patrons in locker rooms become prevalent, the law will eventually catch up to the technology and such practices will be outlawed. Mechanisms to enforce such space-driven configurations need to be developed.

4. Design

Our middleware architecture was first introduced in [11]. We make two main assumptions in our design:

Figure 1: Architecture for automatic device configuration in smart space environments.

1. The smart space environment can infer the social context.
2. The configuration of devices can be standardized. Loosely speaking, we assume a configuration ontology for devices.

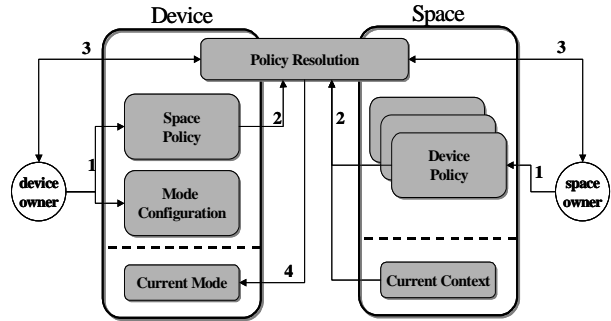


Figure 1 gives a picture of our architecture in which there are two basic entities: devices¹ and spaces. There are 4 phases of interaction:

- Phase 1: **initialization**: device and space are configured by their owners
- Phase 2: **registration**: space and device policies are first examined
- Phase 3: **policy resolution**: conflicts between the device and space policies are resolved, possibly with the interaction of the device and/or space owners
- Phase 4: **configuration**: the device configuration is set

4.1 Devices

Each device type has a standard set of *modes*, which indicates how the device should behave based on the context. For our cell phone example, there might be the following modes:

- **Quiet Mode**: the ringer should be disabled
- **Noisy Mode**: the user is in a loud environment
- **Disable Mode**: all incoming and outgoing calls should be disabled
- **Default Mode**: normal conditions

Other types of devices will have modes that are specific to their functionality. This standard set of modes can loosely be thought of as an ontology, and enables a smart space to “install” a particular type of device once, instead of having to install every possible model of a device².

Each mode can be configured by the user according to limited choices provided by the manufacturer. The manufacturer options should satisfy the mode’s constraints. The quiet mode, for example, could have any incoming call notification mechanism *except* for the ringer. So, the user could choose:

¹ With multimodal devices, we would consider a particular application as an entity instead of a device.

² In reality, devices constantly gain new features leading to different versions of the standardized sets of modes for new generations of devices. Each new version will have to be installed in a space.

- vibrate notification
- flash lights on keypad/screen notification
- take a message and notify later
- play a message telling caller to call back later

The particular configuration that a user chooses for all of the modes is called a device's *mode configuration* and is based purely on users choosing an available configuration for each mode.

The device also has a device's *space policy*, which indicates how a particular space can change the current mode. For example, a user may not want a space to be able to put their phone into disable mode (at least, not without their explicit consent), or a user may completely trust a space they personally manage, such as their home, but want notification whenever public spaces such as a restaurant tries to change their device's mode.

Both the mode configuration and the space policy are determined by the user during phase 1 of Figure 1.

4.2 Spaces

A space maps a social context to device modes. The mapping is called a space's *device policy*. For example, the administrator of a theatre may specify that when a movie is playing, a cell phone device should be in quiet mode. A lecture hall administrator may specify that cell phones be in quiet mode during a lecture and in disable mode during an exam. The device policy is determined by the space owner during phase 1 of Figure 1. While we use the context information in our architecture, we do not address how the space infers this context.

When a device enters a space, it registers with that space. At some point after registration, the device's space policy and the space's device policy must be examined for conflicts, as corresponds to phases 2 in Figure 1. If there are no conflicts, the owners need not be consulted, skipping immediately to phase 4 where the device mode is set. If, however, there is a conflict, the conflict must be resolved. One method to resolve the conflict is to interact with the device and/or space owner, as shown in phase 3 of the figure.

Returning to our example, a user may specify that no space is allowed to put their phone into disable mode. At the same time, a lecture hall specifies that all phones should be disabled during an exam. In such a situation, either the user or the space must relent and this may require user interaction. Initially, the device can interrupt the user and describe the conflict, attempting to get the user to resolve the problem. The user could decide to accept the space's policy always, this one time, or not at all. If the user rejects the space policy, there must be a mechanism to notify the space owner (or current owner, in

this case, the instructor), that a device is not configured properly. Then, it is up to the user and the owner to resolve the problem. Knowing that a particular student's cell phone is not disabled may allow the instructor to monitor that student during the exam more carefully, therefore not requiring the conflict to be resolved in the virtual world.

5. Policy resolution techniques

There are three major issues with policy resolution: *when* the conflicts should be resolved, *how* they should be resolved and *where* the policy resolution logic resides. Our architecture does not attempt to address these issues, but instead, provides enough flexibility that we may explore them. This section describes several approaches that we are beginning to investigate.

The primary issue is how to resolve policy conflict that may arise between the user and the space. We are currently investigating two different approaches: one in which the user must be involved in resolving the conflict, and one in which the resolution is completely automated by agents. Below is a detailed description of both approaches.

5.1 User Intervention

User intervention involves the policy resolution module bringing any policy conflict to the attention of the device owner, and, if necessary, to the attention of the space owner as well. In this approach, the issue of when the conflict is brought to the attention of the user is crucial.

For complex policies, many social contexts may rarely be encountered. An exam, for example, is a relatively rare instance in a lecture hall. Should a potential conflict within the context of an exam be brought to the attention of a student every time she attends class? If it is, the student may eventually accept the space policy out of sheer annoyance, then forget the implications later when the context is actually valid.

Alternatively, if a space changes context frequently, the device may frequently interrupt its owner to resolve configuration problems. Such a high annoyance factor could lead the owner to adopt a lax space policy.

With both of these cases, there runs a risk that the device is automatically configured in ways the owner does not anticipate or want. For automatic configuration to become viable, a balance must be achieved that makes policy resolution manageable. A hybrid approach is for the likely contexts to be resolved when the device enters a space, deferring decisions for contexts that are not likely to occur.

5.2 Agent-based negotiation

The agent-based negotiation approach is used to resolve policy conflicts automatically. Since the introduction of game theory, agent-based negotiation has played a major role in automated conflict resolution [12, 13]. Negotiation is defined as the process by which a group of agents communicate with one another to try and come to a mutually acceptable agreement on some matter [14]. The agent model has been adapted in many Ubicomp projects [15-17] and agent characteristics (autonomy, mobility, intelligence, cooperation, and adaptability) make it very suitable for our purpose as well.

Agent-based negotiation for conflict resolution has been applied to a wide-range of different systems. Air traffic management is one of the prime examples where automated conflict resolution is of great interest [18]. Network management, manufacturing systems, and electronic commerce are other examples where automated agent policy resolution is widely used [19, 20]. Many conflict resolution strategies for multi-agent systems have been developed in the last two decades, including negotiation, voting, and priority convention [21-23]. We are investigating the different strategies and trying to find the one that best fits our system's needs.

The main challenge with automated agent negotiation is to decide the best negotiation strategy given the limitation (computing, battery, and memory) of mobile devices. Another challenge is finding the correct metrics that quantify both the space and device preferences. One possibility we are investigating is to add a weight factor for each mode of the device and the space that reflects the urgency of such mode.

Finally, an important issue for either the user intervention or agent-based negotiation techniques is where the policy resolution logic resides. There are security concerns about a device or space allowing access to their policies by untrusted entities.

One of the major advantages of using agents, in addition to the automation, is their aid in overcoming the trust issue between the space and the users. The trust issue arises because of the need for policy exchange between the device and the space; if either side has knowledge of the other's configuration policy, they could exploit that knowledge maliciously. By using the agent model for negotiation, agents exchange information on a need-to-know basis. Extensive research has been done in the field of agent negotiation between untrusting parties and between non-cooperative environments [13]. We are looking into ways of mapping such techniques to our system.

6. Current Status and Future Plans

We have implemented the user-intervention part of the system and are currently working on the agent-based part. Our prototype runs on a Tungsten W smart phone that is running Palm OS 4.1.1. We would like to have both prototypes available before we start the usability testing process.

Anywhere automation is involved, the question of user control arises. The issue weighs the convenience of automation against the degree of control over one's devices the user is willing to relinquish. Shneiderman pointed out that in most cases, humans like to be in control of their actions and interactions [24]. However, new research has suggested that people are willing to give up part of the control in return for convenience [25]. However, there are no long term usability studies that definitively support either viewpoint. We believe automatic device configuration can provide a positive experience for the user; however, providing the right balance of control versus convenience is essential.

As such, in order to make sure our goal of minimizing user distraction and thus increasing user convenience is achieved, we are planning long term user studies with our PDA/cell phone prototype.

For our agent-based negotiation techniques, we plan to investigate the feasibility of different criteria having different priorities. For example, legal use is always enforced, followed by maintaining security of devices and spaces, followed by achieving acceptable, if not ideal, configuration for participating parties.³

As the number of devices and the number of contexts grow, it is likely that a device will not behave as expected by the user all of the time. A major cause of unintended side effects is that the device owner and space owner will most often be different people with different goals and models of use. It is important that we recognize the likelihood of unanticipated behavior; and while we will strive to reduce their frequency, we must develop mechanisms for users to determine the reason a device is configured a certain way and the ability to make corrective actions for the future.

Finally, while our architecture can easily suggest configuration modes to a device, the issue of enforcement is still open. Certain devices from manufacturers can be designated as "compliant". But, just as banning video phones won't get rid of regular cameras, adopting our infrastructure will not rid the world of non-compliant video phones. And while the cell phone manufacturers could adopt an industry-wide standard, there are many types of devices that are programmable, such as PDAs, for which a simple compliance tag will not suffice.

³ This approach is somewhat reminiscent of Isaac Asimov's Three Laws of Robotics as presented in *I, Robot*.

Ultimately, a device owner can program certain devices to appear to be compliant, when in reality, they are not.

7. Concluding remarks

This work was inspired by the recent advances in smart spaces and by the introduction of many new smart mobile devices and gadgets. By having automatic device configuration we hope to minimize user distraction and can get closer to achieving the goals of ubiquitous computing. We have presented a framework for automatic device configuration for smart spaces, and discussed the issue involved in negotiating a configuration based on user preferences and the space's policy. We described two approaches to configuration negotiation that we are exploring: user intervention and agent-based negotiation. While we are using a cell phone scenario as our guiding example, our framework and implementation can be generalized for any mobile smart device.

Aside from our implementation efforts, we envision several areas for future work. These involve extensive user studies to determine the best balance of control vs. convenience, providing useful feedback to the user when they encounter unanticipated behavior, and investigating enforcement for malicious devices.

Acknowledgments

This work was partially supported by a grant from the Lilly Endowment and the Center for Applied Cybersecurity Research at Indiana University.

9. References

- [1] Popyack, J., et al. "Academic Dishonesty in a High-Tech Environment", In Special Session at the 34th SIGCSE Technical Symposium on Computer Science Education, 2003, Reno, Nevada.
- [2] "Camera-equipped cell phones spread new brands of mischief", 10 July, 2003, http://www.usatoday.com/tech/news/2003-07-10-cell-mischief_x.htm.
- [3] Kamijoh, N., et al. "Linux Watch: Hardware Platform for Wearable Computing Research", In Second IEEE Pacific-RIM Conference on Multimedia, 2001, Beijing.
- [4] Want, R., et al., "The Active Badge Location System", ACM Transactions on Information Systems, 1992.
- [5] Popovici, A., A. Frei, and G. Alonso. "A Proactive Middleware Platform for Mobile Computing", In Proc. of the 4th International Middleware Conference, 2003, Rio de Janeiro, Brazil.
- [6] Kon, F., et al. "Dynamic Resource Management and Automatic Configuration of Distributed Component Systems", In Proceedings of the 6th USENIX Conference on Object-Oriented Technologies and Systems, 2001, San Antonio, Texas.
- [7] Fox, A., et al., "Integrating Information Appliances into an Interactive Workspace", IEEE Computer Graphics and Applications, 2000: p. 54-65.
- [8] Román, M., et al., "Gaia: A Middleware Infrastructure to Enable Active Spaces", IEEE Pervasive Computing, 2002: p. 74-83.
- [9] Siewiorek, D., et al. "SenSay: A Context-Aware Mobile Phone", Submitted to the International Symposium on Wearable Computers, 2003.
- [10] Schmidt, A., A. Takaluoma, and J. Mntyjrvi. "Context-Aware Telephony over WAP", Presented to Handheld and Ubiquitous Computing (HUC2k), 2000: Springer-Verlag, London, Ltd.
- [11] Connelly, K. and A. Khalil. "Towards Automatic Device Configuration in Smart Environments", In Proceedings of UbiSys Workshop, 2003.
- [12] Rosenschein, J.S. and G. Zlotkin, "Rules of Encounter: Designing Conventions for Automated Negotiation among Computers", 1994, Cambridge Massachusetts: MIT Press.
- [13] Zlotkin, G. and J. Rosenschein, "Cooperation and Conflict Resolution via Negotiation among Autonomous Agents in Non Cooperative Domains", IEEE Transactions on Systems, Man and Cybernetics, 1991, 21(6): p. 1317-1324.
- [14] Jennings, N.R., et al. "Automated Negotiation", In Proc. 5th Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Systems (PAAM-2000), Manchester, UK.
- [15] Ranganathan, A. and R.H. Campbell. "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments", In Middleware, 2003.
- [16] Hristova, N., G.M.P. O'Hare, and T. Lowen. "Agent-based Ubiquitous Systems: 9 Lessons Learnt", In UbiSys-Workshop at the Fifth Annual Conference on Ubiquitous Computing, 2003, Seattle.
- [17] Muldoon, C., et al. "ACCESS: An Agent Architecture for Ubiquitous Service Delivery", In Proceedings Seventh International Workshop on Cooperative Information Agents (CIA2003), 2003, Helsinki.
- [18] Tomlin, C., G.J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multi-agent hybrid systems", IEEE Trans. Automat. Contr., 1998, 43: p. 509-521.

- [19] Skarneas, N. and K.L. Clark. "Process Oriented Programming for Agent Based Network Management", In Proceedings of ECAI Workshop on Intelligent Agents for Telecommunications Applications (IATA96), 1996: IOS Press.
- [20] Limthanmaphon, B., Y. Zhang, and Z. Zhang. "An agent-based negotiation model supporting transactions in Electronic Commerce", In IEEE 11th International Workshop on Database and Expert Systems Applications, 2000.
- [21] Sycara, K. "Resolving Goal Conflict via Negotiation", In The Seventh National Conference on Artificial Intelligence, 1988.
- [22] Ephrati, E. and J.S. Rosenschein. "The Clarke Tax as a Consensus Mechanism Among Automated Agents", In Proceedings of the Ninth Conference on Artificial Intelligence, 1991.
- [23] Ioannidis, Y.E. and T.K. Sellis. "Conflict Resolution of Rules Assigning Values to Virtual Attribute", In Proceedings of the 1989 ACM International Conference on the Management of Data, 1989, Portland, OR.
- [24] Shneiderman, B., "Designing the User interface: Strategies for Effective Human-Compute Interaction", 3rd ed, 1998, Reading, MA: Addison-Wesley.
- [25] Barkhuus, L. and A.K. Dey. "Is context-aware computing taking control away from the user? Three levels of interactivity examined", In UBICOMP 2003, 5th International Symposium on Ubiquitous Computing, 2003.